Questions

Before starting test Please Refer the Document named "Base Program.py" and executes to generate a dictionary.

#Program 1

Use cases are a cycle or iteration starts with a key and ends with the same key but should not consider at last in the dictionary.

1. From the generated dictionary Sort the keys in ordered list.
2. Iterate the keys of the list, Check the cycle for each key from the sorted list.

Example:-

Generated dictionary mapping with inputs `10 9`

```
{1: 5, 2: 6, 3: 5, 4: 5, 5: 6, 6: 7, 7: 1, 9: 6}
```

The result cycles by the mapping are:

```
[1, 5, 6, 7]
```

Use Cases:

1. Generate a cycle from the mapping with `0 11`

   Expected Output:-

   ```
   [[2, 7], [10]]
   ```

2. Generate a cycle from the mapping with `12 38`

   Expected Output:-

   ```
   [[4, 38, 37, 22, 33, 9], [11, 36, 31]]
   ```

3. Generate a cycle from the mapping with `0 6`

   Expected Output:-

   ```
   [[1], [3], [6]]
   ```

4. Generate a cycle from the mapping with `20 11`

   Expected Output:-

   ```
   [[4], [5, 8, 11], [10]]
   ```

5. Generate a cycle from the mapping with `50 15`

   Expected Output:-

   ```
   [[5], [9, 10, 15]]
   ```

6. Generate a cycle from the mapping with `34 56`

   Expected Output:-

   ```
   [[1, 34, 12, 38, 55, 48, 35], [4, 11], [6, 47], [41]]
   ```

#Program 2

Triply Ordered reverse a dictionary per lengths.

1. From the generated dictionary Sort the keys in ordered list.
2. Iterate the keys of the list and prepare the cycles per length.
3. No Key appears more than one but the values can be duplicated.
4. Wherever the key is the value that occurrences will be list of values.

Example:-

Generate a new dictionary `10 11` as the inputs

```
{1: 5, 2: 11, 3: 6, 4: 11, 5: 10, 6: 5, 7: 5, 8: 6, 9: 11,
10: 7, 11: 1}
```

Result (triply ordered) reverse dictionary per length

```
{1: {1: [11], 7: [10], 10: [5]},
 2: {6: [3, 8]},
 3: {5: [1, 6, 7], 11: [2, 4, 9]}}
```

Use Cases:

1. Generate a new (**triply ordered**) dictionary `0 4` as the inputs

   Expected output:-

   ```
   {1: {1: [4], 3: [2]}}
   ```

2. Generate a new (**triply ordered**) dictionary `0 6` as the inputs

   Expected output:-

   ```
   {1: {1: [1], 3: [3]}, 2: {6: [5, 6]}}
   ```

3. Generate a new (**triply ordered**) dictionary `20 11` as the inputs

   Expected output:-

   ```
   {1: {1: [9], 2: [6], 8: [5], 9: [3], 10: [10], 11: [8]},
   2: {4: [2, 4], 5: [7, 11]}}
   ```

4. Generate a new (**triply ordered**) dictionary `50 15` as the inputs

   Expected output:-

   ```
   {1: {3: [4], 6: [8], 8: [14], 9: [15], 10: [9], 12:
   [11]}, 2: {14: [2, 13]}, 3: {5: [1, 5, 6]}, 4: {15: [3,
   7, 10, 12]}}
   ```

5. Generate a new (**triply ordered**) dictionary `12 38` as the inputs

   Expected output:-

   ```
   {1: {8: [16], 12: [24], 17: [32], 23: [28], 26: [34], 29:
   [15], 30: [35], 31: [36], 33: [22]}, 2: {4: [9, 25], 6:
   [5, 12], 13: [2, 17], 36: [6, 11], 37: [8, 38]}, 3: {3:
   [13, 20, 30], 9: [7, 10, 33], 11: [1, 27, 31], 22: [19,
   29, 37], 38: [3, 4, 21]}}
   ```

6. Generate a new (**triply ordered**) dictionary `34 56` as the inputs

   Expected output:-

   ```
   {1: {6: [47], 11: [4], 13: [28], 17: [54], 22: [23], 24:
   [7], 25: [43], 27: [9], 28: [5], 29: [33], 34: [1], 39:
   [17], 40: [52], 42: [40], 43: [53], 47: [6], 50: [44],
   52: [49], 53: [30], 56: [45]}, 2: {2: [25, 32], 8: [2,
   36], 12: [26, 34], 33: [24, 39], 41: [41, 56], 48: [37,
   55]}, 3: {1: [29, 35, 51], 4: [11, 15, 50], 38: [10, 12,
   31], 55: [16, 20, 38]}, 4: {35: [3, 19, 27, 48]}}
   ```